

PLP - Recuperatorio del Primer Parcial - 1^{er} cuatrimestre de 2023

Este examen se aprueba obteniendo al menos dos ejercicios bien menos (B-) y uno regular (R). Las notas para cada ejercicio son: -, I, R, B-, B. Entregar cada ejercicio en hojas separadas. Poner nombre, apellido y número de orden en todas las hojas, y numerarlas. Se puede utilizar todo lo definido en las prácticas y todo lo que se dio en clase, colocando referencias claras. El orden de los ejercicios es arbitrario. Recomendamos leer el parcial completo antes de empezar a resolverlo.

Ejercicio 1 - Programación funcional

Considere la siguiente representación para naves espaciales, donde dos naves pueden fusionarse a través de un módulo conector para formar una nave más grande y potente:

`data Componente = Contenedor | Motor | Escudo | Cañón deriving Eq`

`data NaveEspacial = Módulo Componente NaveEspacial NaveEspacial | Base Componente deriving Eq`

Definir las siguientes funciones:

- a. `recNave` y `foldNave`, los esquemas de recursión primitiva y estructural respectivamente. Indicar los tipos de ambos esquemas.

Para la definición de `recNave` se puede utilizar recursión explícita. En cambio `foldNave` deberá definirse a partir de `recNave`.

Nota: en los puntos que siguen se deberá utilizar el esquema más adecuado para cada caso.

- b. `espejo :: NaveEspacial -> NaveEspacial`, que dada una nave devuelve otra con los mismos componentes, pero en posiciones espejadas (los de la derecha a la izquierda y viceversa).
- c. `esSubnavePropia :: NaveEspacial -> NaveEspacial -> Bool` que, dadas dos naves, indica si la primera está contenida propiamente dentro de la segunda. Solo se considerarán subnaves propias aquellas que se encuentren al fondo. Es decir, que se encuentren en la segunda nave como "subárboles" sin otros componentes más abajo, y que no sean iguales a la segunda nave.
- d. `truncar :: NaveEspacial -> Integer -> NaveEspacial`, que dadas una nave y un número natural n devuelva una nave con los niveles 0 a n de la original, siendo el nivel 0 la raíz.

Ejercicio 2 - Inferencia

Se extiende el cálculo lambda con el tipo $BDD_{\sigma, \tau}$, que representa un árbol de decisiones binarias que, al evaluarse en valores de tipo σ , arrojará resultados de tipo τ . Es decir, un árbol binario no vacío cuyos nodos internos contienen predicados (funciones que devuelven valores de tipo `Bool`), los cuales se podrán evaluar en un valor dado para definir por qué rama descender hasta llegar a una hoja, la cual contiene el resultado final. ($BDD = Binary Decision Diagram$).

En este ejercicio nos concentraremos en la estructura del BDD y no en su evaluación.

$\sigma ::= \dots \mid BDD_{\sigma, \tau} \quad M ::= val_{\sigma}(M) \mid M ? M : M \mid case\ M\ of\ val(x) \rightsquigarrow M; p ? i : d \rightsquigarrow M \quad \text{donde:}$

- $BDD_{\sigma, \tau}$ es el tipo de los diagramas de decisiones binarias que admiten valores de tipo σ y arrojan resultados de tipo τ .
- $val_{\sigma}(M)$ representa un diagrama sin predicados, que al evaluarse en un valor de tipo σ siempre arroja como resultado M - o, más precisamente, el resultado de reducir M a un valor.
- $M_1 ? M_2 : M_3$ representa al diagrama cuya raíz es la función M_1 , y sus subárboles son M_2 y M_3 . Si al evaluar el diagrama para un valor V la aplicación de M_1 a V devuelve `True`, se continuará evaluando el subárbol M_2 . En caso contrario, se descenderá por el subárbol M_3 .
- $case\ M_1\ of\ val(x) \rightsquigarrow M_2; p ? i : d \rightsquigarrow M_3$ representa al observador `case` del diagrama M_1 , donde la variable x puede estar libre en M_2 y las variables p, i y d pueden estarlo en M_3 .

Las reglas de tipado son las siguientes:

$$\frac{\Gamma \triangleright M : \tau}{\Gamma \triangleright \text{val}_\sigma(M) : \text{BDD}_{\sigma,\tau}} \text{(T-VAL)} \quad \frac{\Gamma \triangleright M_1 : \sigma \rightarrow \text{Bool} \quad \Gamma \triangleright M_2 : \text{BDD}_{\sigma,\tau} \quad \Gamma \triangleright M_3 : \text{BDD}_{\sigma,\tau}}{\Gamma \triangleright M_1 ? M_2 : M_3 : \text{BDD}_{\sigma,\tau}} \text{(T-CHOICE)}$$

$$\frac{\Gamma \triangleright M_1 : \text{BDD}_{\sigma,\tau} \quad \Gamma, x : \tau \triangleright M_2 : \rho \quad \Gamma, p : \sigma \rightarrow \text{Bool}, i : \text{BDD}_{\sigma,\tau}, d : \text{BDD}_{\sigma,\tau} \triangleright M_3 : \rho}{\Gamma \triangleright \text{case } M_1 \text{ of } \text{val}(x) \rightsquigarrow M_2; p ? i : d \rightsquigarrow M_3 : \rho} \text{(T-CASEBDD)}$$

- a) Extender el algoritmo de inferencia para soportar la nueva extensión.
- b) Aplicar el algoritmo extendido con el método del árbol para dar el tipo de la siguiente expresión, exhibiendo las sustituciones utilizadas. De no tipar, indicar el motivo.

$\text{case } f ? x : y \text{ of } \text{val}(x) \rightsquigarrow x; x ? y : z \rightsquigarrow f \ 0$

Ejercicio 3 - Subtipado

Se extiende el cálculo lambda con el tipo $\text{Competencia}(\sigma)$, que permite modelar competencias entre participantes de tipo σ . Una competencia consta de una serie de desafíos por los que pasarán sucesivamente dos competidores, en cada uno de los cuales puede ganar el primer participante, el segundo o puede haber un empate. El ganador será el participante que haya superado más desafíos, o empate en caso de que ambos hayan ganado la misma cantidad.

Los conjuntos de tipos y términos se extienden de la siguiente manera:

$\sigma ::= \dots \mid \text{Resultado} \mid \text{Competencia}(\sigma)$

$M ::= \text{Primero} \mid \text{Segundo} \mid \text{Empate} \mid \text{Desafío}_{x,y} \succ M_1 ++ M_2 \mid \text{Final}_\sigma \mid \text{Competir } M \text{ contra } M \text{ en } M \text{ donde:}$

- $\text{Competencia}(\sigma)$ es el tipo de las competencias con participantes de tipo σ , y Resultado el de sus resultados parciales y finales.
- Primero , Segundo y Empate son los posibles resultados de un desafío y de una competencia.
- $\text{Desafío}_{x,y} \succ M_1 ++ M_2$ representa el agregado de un desafío M_1 a una competencia M_2 , donde M_1 es un término que puede tener libres a las variables x e y , las cuales se ligarán a los participantes una vez iniciada la competencia.
- Final_σ representa el final de una competencia con participantes de tipo σ . Es decir, una competencia que no tiene desafíos por completar.
- $\text{Competir } M_1 \text{ contra } M_2 \text{ en } M_3$ representa el desarrollo de la competencia M_3 entre los participantes M_1 y M_2 . El valor de esta expresión será el resultado de la competencia al completarse todos sus desafíos.

Las reglas de tipado son las siguientes:

$$\frac{}{\Gamma \triangleright \text{Primero} : \text{Resultado}} \text{(T-PRI)} \quad \frac{}{\Gamma \triangleright \text{Segundo} : \text{Resultado}} \text{(T-SEG)} \quad \frac{}{\Gamma \triangleright \text{Empate} : \text{Resultado}} \text{(T-EMP)}$$

$$\frac{\Gamma, x : \sigma, y : \sigma \triangleright M_1 : \text{Resultado} \quad \Gamma \triangleright M_2 : \text{Competencia}(\sigma)}{\Gamma \triangleright \text{Desafío}_{x,y} \succ M_1 ++ M_2 : \text{Competencia}(\sigma)} \text{(T-DESA)} \quad \frac{}{\Gamma \triangleright \text{Final}_\sigma : \text{Competencia}(\sigma)} \text{(T-FIN)}$$

$$\frac{\Gamma \triangleright M_1 : \sigma \quad \Gamma \triangleright M_2 : \sigma \quad \Gamma \triangleright M_3 : \text{Competencia}(\sigma)}{\Gamma \triangleright \text{Competir } M_1 \text{ contra } M_2 \text{ en } M_3 : \text{Resultado}} \text{(T-COMPETIR)}$$

- a) Dar la(s) regla(s) de subtipado para competencias y justificar en términos del principio de substitutividad.

- b) Utilizando las reglas de tipado y subtipado, derivar el siguiente juicio de tipado:

$\emptyset \triangleright (\lambda c : \text{Competencia}(\text{Nat}).$

$\text{Competir } 0 \text{ contra True en } \text{Desafío}_{x,y} \succ \text{if isZero}(x) \text{ then Primero else Segundo} ++ c) \text{ Final}_{\text{Int}} : \text{Resultado}$